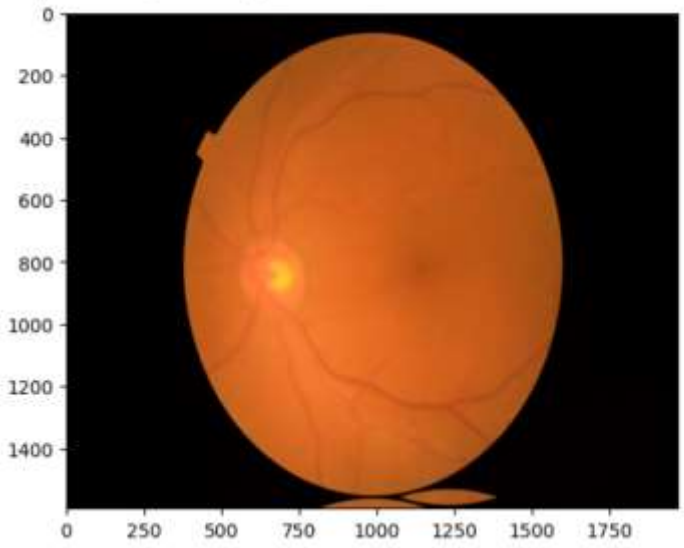
Data Collection and Preprocessing Phase

Date	10th July 2024
Team ID	SWTID1719999219
Project Title	Crystal Clear Vision: Revolutionizing Cataract Prediction through Transfer Learning Mastery
Maximum Marks	6 Marks

Preprocessing Template

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	**Description**
Data Overview	The retina dataset we chose for our project contains four categories: 1) normal 2) cataract 3) glaucoma 4) retina disease. However as only the first two categories were applicable to our project, some preprocessing steps were applied. The unnecessary categories were removed. Next, train, validation and test folders were created for normal and cataract. Totally there were 300 normal eye images and 100 cataract eye images. The split which was going to be used for train, validation and test was 80-10-10 respectively.
Resizing	Resize images to a specified target size. As we are using ResNet50, resizing to 224x224x3 would be applied.
Normalization	Normalize pixel values to a specific range. ResNet50 applies normalization after convolution layers.

Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing. Ex: Image before augmenting:  Image after augmenting: 
Denoising	Apply denoising filters to reduce noise in the images. N/A
Edge Detection	Apply edge detection algorithms to highlight prominent edges in the images. N/A
Color Space Conversion	Convert images from one color space to another.

	N/A (Images are already in RGB format).
Image Cropping	Crop images to focus on the regions containing objects of interest. N/A (Images in the dataset have already been cropped).
Batch Normalization	Apply batch normalization to the input of each layer in the neural network. ResNet50 applies batch normalization after convolution layers.
Data Preprocessing Code Screenshots	

Loading Data	**Downloading from Kaggle:**
	Downloading dataset from kaggle
	`!mkdir -p ~\.kaggle`
	`!cp kaggle.json ~/.kaggle`
	`!kaggle datasets download -d jr2ngb/cataractdataset`
	`Dataset URL: https://www.kaggle.com/datasets/jr2ngb/cataractdataset`
	`License(s): unknown`
	`Downloading cataractdataset.zip to /content`
	`100% 3.34G/3.34G [03:21<00:00, 16.4MB/s]`
	`100% 3.34G/3.34G [03:21<00:00, 17.8MB/s]`
	`!unzip '/content/cataractdataset.zip'`
	
	Remove unnecessary directories:

<table>
<tr>
<td></td>
<td>

```
#to remove unnecessary datasets
# Define the path to the folder containing the classes
path = '/content/repository/yiweichen04-retina_dataset-914b0f4/dataset'

# List all directories and store them in a Python list
dirs = !ls -d $path/*/

# Convert the paths to a list of strings
dirs = [d.strip() for d in dirs]

# Print the directories to be removed
print(f"Directories to be removed: {dirs[-2]}, {dirs[-1]}")

# Remove the last two directories
!rm -r {dirs[-2]}
!rm -r {dirs[-1]}

# Verify the removal
!ls $path
```

Loading data from train and validation directories:

```
import tensorflow as tf

train_dir='/content/repository//yiweichen04-retina_dataset-914b0f4/train'
val_dir='/content/repository/yiweichen04-retina_dataset-914b0f4/validation'

img_size=(224,224)
train_data=tf.keras.preprocessing.image_dataset_from_directory(directory=train_dir,
                                                               label_mode='binary',
                                                               image_size=img_size,
                                                               batch_size=32)
val_data=tf.keras.preprocessing.image_dataset_from_directory(directory=val_dir,
                                                             batch_size=32,
                                                             label_mode='binary',
                                                             image_size=img_size)
```

</td>
</tr>
<tr>
<td>Resizing</td>
<td>

```
import tensorflow as tf

train_dir='/content/repository/yiweichen04-retina_dataset-914b0f4/train'
val_dir='/content/repository/yiweichen04-retina_dataset-914b0f4/validation'

img_size=(224,224)
train_data=tf.keras.preprocessing.image_dataset_from_directory(directory=train_dir,
                                                               label_mode='binary',
                                                               image_size=img_size,
                                                               batch_size=32)
val_data=tf.keras.preprocessing.image_dataset_from_directory(directory=val_dir,
                                                             batch_size=32,
                                                             label_mode='binary',
                                                             image_size=img_size)
```

</td>
</tr>
<tr>
<td>Normalization</td>
<td>Normalization is done after augmentation in this line:

```
augmented_img= tf.squeeze((augmented_img)/255.)[:, :, ::-1] #normalize after augmentati
```

</td>
</tr>
</table>

	3 conv1_bn 4 conv1_relu 5 pool1_pad 6 pool1_pool 7 conv2_block1_1_conv 8 conv2_block1_1_bn 9 conv2_block1_1_relu 10 conv2_block1_2_conv 11 conv2_block1_2_bn 12 conv2_block1_2_relu 13 conv2_block1_0_conv 14 conv2_block1_3_conv 15 conv2_block1_0_bn 16 conv2_block1_3_bn 17 conv2_block1_add 18 conv2_block1_out The layers with suffix 'bn' in ResNet50 signify batch normalization.

Data Augmentation	
	`from tensorflow.keras import layers`
	`import tensorflow as tf`
	
	`#we are going to create a data augmentation layer to be used later with the model.`
	`data_augmentation= tf.keras.Sequential([`
	`layers.RandomFlip("horizontal"),`
	`layers.RandomRotation(0.2),`
	`layers.RandomZoom(0.2),`
	`layers.RandomHeight(0.2),`
	`layers.RandomWidth(0.2)`
	
	`], name="data_augmentation")`
	
	`#now we will test this layer.`
	`path="/content/repository/yiweichen04-retina_dataset-914b0f4/dataset/2_cataract/cataract_075.png"`
	`img_show(path,"image before augmenting")`
	`img=cv2.imread(path)`
	`augmented_img= data_augmentation(tf.expand_dims(img,axis=0))`
	`augmented_img= tf.squeeze((augmented_img)/255.)[:, :, ::-1] #normalize after augmentation`

Denoising	Give the code snippet as an image (copy and paste the picture in this block). N/A
Edge Detection	Give the code snippet as an image (copy and paste the picture in this block). N/A
Color Space Conversion	Give the code snippet as an image (copy and paste the picture in this block). N/A

Image Cropping	Give the code snippet as an image (copy and paste the picture in this block). N/A
Batch Normalization	Give the code snippet as an image (copy and paste the picture in this block). 3 conv1_bn 4 conv1_relu 5 pool1_pad 6 pool1_pool 7 conv2_block1_1_conv 8 conv2_block1_1_bn 9 conv2_block1_1_relu 10 conv2_block1_2_conv 11 conv2_block1_2_bn 12 conv2_block1_2_relu 13 conv2_block1_0_conv 14 conv2_block1_3_conv 15 conv2_block1_0_bn 16 conv2_block1_3_bn 17 conv2_block1_add 18 conv2_block1_out The layers with suffix ‘bn’ in ResNet50 signify batch